Celtrin izziv - predstavitev

Uvod

Sem Marko Grešak, izdelal sem HTML5 igro v sklopu Celtrinega programerskega izziva.

Več o meni: naslov moje strani, GitHub ter Twitter.

Igro lahko igrate na gresak.io/celtra, od koder ste preusmerjeni na server z igro.

Pa še to: prosojnice sem iz navade naredil v angleščini.

(naprej)

O igri

Kot zahtevano pri izzivu, **večigralna**, tema pa je bojevanje igralcev med seboj.

(naprej)

Igralci igrajo na naključno generirani platformi, ki pa je, jasno, identična za vse igralce. Platforma se generira glede na ime igre, ki je sicer nastavljeno direktno v kodi, ideja pa je bila, da imajo igralci možnost izbira sobe v kateri igrajo, kjer je ime sobe uporabljeno kot ime igre. To pomeni, da bi ime sobe vplivalo na to, kako zgleda platforma.

(naprej)

Kako igrati:

- Premik levo s tipko `a` ali `puščico levo`
- Premik desno s tipko `d` ali `puščico desno`
- Napadanje drugi igralcev ko smo v bližini `presledek`

Ker pa na **mobilnih napravah** tipično ne igramo s tipkovnicami, uporabimo:

- *(naprej)*
- dotik na ekran za premik levo oziroma desno
- pritisnemo gumb Attack, ki je viden samo na mobilnih napravah

(naprej)

Prijavno okno

Na prijavnem oknu je vnosno polje za uporabniško ime, s katerim bo uporabnik predstavljen v igri.

(naprej)

Igralno okno

Modro ozadje igre je nastavljeno v CSS kot ozadje containerja canvasa.

(naprej)

Na canvas se najprej izriše zelena platforma.

(naprej)

Nato je zgoraj levo prikazan nivo zdravja, ki ima širino do največ polovico zaslona.

(naprej)

Nazadnje pa se izriše še igralec in morebirni ostali igralci.

(naprej)

Pogled na mobilnih napravah

Pogled na mobilnih napravah je identičen tistemu na zasebnih računalnikih, z razliko *(naprej)* gumba za napad, ki je prikazan v spodnjem levem kotu.

(naprej)

Tehnologije

Kot je zahteval izziv, projekt sledi HTML5 standardom, uporabil pa sem HTML5 `canvas` element za prikaz igre. Stran je oblikovana z osnovnim CSSom, vsebuje pa tudi nekaj CSS3 selektorjev, primer je onemogočena izbira besedila, ki je lahko nadležna za mobilne uporabnike.

Za programiranje logike na client in server delu pa sem izbral *(naprej)* **Dart**.

(naprej)

Dart?

(naprej)

Je eden od "**transpile to JS**" jezikov oziroma je jezik, ki se prevede v JavaScript. Zelo dobro rešuje probleme, na katere naletimo pri delu z JavaScriptom, ter dodaja funkcionalnosti, ki jih JavaScript (še) nima. *(naprej)*

Vsebuje **vsa orodja** in veliko knjižnic, ki jih potrebujemo za delo. To vključuje web API skupaj z WebSocketi, IO API na strežnikih, ki vključuje med drugim tudi strežniški del WebSocketov. Omogoča tudi enostavno delo s frameworkom Angular ter knjižnico Polymer.

(naprej)

Dartov SDK vsebuje vsa orodja za razvoj projekta. Že ko namestimo dart, dobimo celoten SDK z vsemi orodji, skupaj z IDEjem Dart Editor. Orodja omogočajo serviranje datotek iz `web/` datoteke, namestitev novih modulov, objavljanje modulov na dartov repozitorij, analizo projekta, generiranje dokumentacije in drugo.

(naprej)

Za razliko od ostalih jezikov s podobnim namenom, za Dart obstaja **`Dart VM`**, ki deluje tudi v brskalnikih. Trenutno deluje samo v brskalniku `Dartium`, ki je prilagojena verzija brskalnika Chromium. Google planira, da bo Dart kmalu podprt tudi v brskalnikih Chrome. Za zdaj je za podporo ostalim brskalnikom še vedno potrebno pretvoriti Dart v JavaScript z ukazom `dart2js`.

(naprej)

Dart je podprt tudi v serverskih okoljih kot **`Standalone VM`**, ki podpira vse knjižnice, ki niso povezane s HTML APIjem.

(naprej)

Deployanje Dart projektov

Kako objavimo projekt za testiranje ali produkcijo.

(naprej)

Kot že omenjeno, ima Dartov SDK že vključeno **orodje za serviranje** projektov.

(naprej)

To orodje se imenuje **`pub serve`** in servira datoteke, ki se nahajajo v `web/` direktoriju. Lahko mu podamo tudi na katerem naslovu in portu naj posluša, privzeto pa je to `localhost` na portu `8080`. Če strežnik zazna, da odjemalec ne podpira Darta, po potrebi compila, potem pa vrne JavaScript verzijo kode. Ta ukaz uporabljam za zagon client-side dela igre.

(naprej)

Za zaganjanje pozameznih datotek, npr. serverja, pa lahko uporabimo kar ukaz `dart` in mu podamo ime datoteke. Ta ukaz ukaz uporablam za zagon WebSocket strežnika.

(naprej)

Za **ločitev strežnika** sem se odločil zaradi *(naprej)* lažje skalabilnosti projekta. Sicer je projekt daleč od tega, da bi zahteval skalabilnost, namen je bil implementirati to idejo v projekt.

(naprej)

Kako deluje: Client

Kratka predstavitev delovanja client-side dela.

(naprej)

Začne igro z vnešenim uporabniškim imenom.

(naprej)

Poveže se na strežnik preko `WebSocket` razreda znotraj `dart:html`.

(naprej)

Naključno generira platformo.

(naprej)

Ob vsakem premiku igralca pošlje posodobitev na server.

(naprej)

Kako deluje: Server

(naprej)

Začne poslušati zahteve na naslovu in portu, v oddani izvorni kodi je to `localhost` na portu `9999`.

(naprej)

Povezava s clienti poteka preko `WebSocket` razreda znotraj `dart:io`.

(naprej)

Ob prejemu zahtevka ga pretvori v sporočilo in opravi definirano akcijo, npr. dodajanje novega igralca.

(naprej)

Vsem igralcem, razen pošiljatelju, pošlje prejeto sporočilo.

(naprej)

Komentarji, vprašanja

(naprej)

Zahvaljujem se vam za vašo pozornost. Še enkrat povezava na igro in naslov, kjer lahko najdete to predstavitev.